843161-103

# SYSTEM AND METHOD FOR SMALL READ ONLY DATA

## BACKGROUND OF THE INVENTION

### 1. Field of the Invention

The present invention relates to minimizing data retrieval time and/or area, or more particularly, to a system and method for minimizing read-only data retrieval time and/or area by using combinatorial logic to provide read-only data to a requester.

### 2. Description of Related Art

Electronic circuits typically operate in conjunction with at least one form of memory. For example, a personal computer generally includes a central processing unit (CPU) that operates in conjunction with random-access memory (RAM) and read-only memory (ROM) -- the major difference being that RAM can both store and retrieve user defined data whereas ROM can only retrieve data established at the time of manufacture.

Typically, RAM is constructed by dividing a read/write portion of memory into discrete address bits, thus allowing data to be stored at, and retrieved from, a particular address location. For example, in response to a computer user providing a data-processing program (e.g., Word, Word Perfect, etc.) with the word "dog," the program might temporarily store the word in RAM at a particular address location (e.g., address one-thousand). If the document is subsequently printed, the program will retrieve the word "dog" from the previously stored address location (e.g., address one-thousand) and provide it to the designated printer.

As well, ROM, regardless of its limited use (in that it cannot store user provided data), is typically constructed in the same fashion. A read-only portion of memory is divided into discrete address bits to allow read-only data to be retrieved from a particular address location. For example, a Pentium processor, which uses load

constant instructions, may retrieve these constants from a particular address location in ROM -- these constants being established at known addresses in the ROM module at the time of manufacturing.

The benefit of similar construction is that all forms of data -- both stored and manufacturer established (i.e., read-only) data -- can be retrieved by merely providing the memory module with the address location at which the data is located. The disadvantage of using such a similar construction is that both memory modules take roughly the same amount of time to execute (i.e., roughly an entire clock cycle), and utilize the same amount of real estate to store a single bit of data. In other words, ROM, which is more simplistic (in terms of read/write) than RAM, takes approximately the same amount of time to execute an instruction (i.e., provide data), and uses the same amount of area (per bit), as RAM.

This becomes especially problematic when the read-only memory (ROM) is small in comparison to the random-access memory (RAM), because a relatively large amount of time (usually an entire clock cycle) is needed to differentiate between a relatively small amount of manufacturer established data (i.e., the read-only data stored in ROM). Thus it would be desirable to have a system and method that minimizes read-only data retrieval time and/or area for ROM devices.

## SUMMARY OF THE INVENTION

The present invention provides a system and method for minimizing read-only data retrieval time through the use of combinatorial logic. One embodiment of the present invention is directed to a two-bit address system. In this embodiment, two address bits are provided to a binary logic function device. The binary logic function device uses the two address bits and predetermined logic functions (i.e., functions that represent a plurality of read-only data values) to produce binary values. The resulting binary value is the read-only data requested.

Another embodiment of the present invention is directed to a three-bit address system. In this embodiment, two of the address bits are used as in the first embodiment, with the binary values (resulting from the binary logic function device)

being provided to a multiplexer. The multiplexer uses the remaining address bit to select which binary value is the read-only data that has been requested.

Another embodiment of the present invention is directed to a multi-bit address systems (e.g., a six-bit address systems). In this embodiment, the binary values (resulting from the binary logic device) are provided to at least one multiplexer (the number of multiplexers and/or their ratios being determined by the number of address bits available as well as the characteristics of the miltiplexers available to the designer). The multiplexer uses a portion of the remaining bits (i.e., the address bits not being provided to the binary logic function device) to select (or narrow down) which binary value is the read-only data that has been requested. These narrowed down values are then provided to at least one other multiplexer -- where the remainder of the remaining bits are used to select which value is the read-only data that has been requested.

A more complete understanding of the system and method for minimizing read-only data retrieval time by using combinatorial logic will be afforded to those skilled in the art, as well as a realization of additional advantages and objects thereof, by a consideration of the following detailed description of the preferred embodiment. Reference will be made to the appended sheets of drawings which will first be described briefly.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 depicts the address combinations of a two-bit address system.

Fig. 2 depicts sixteen binary logic functions (i.e., a-q) available for a two-bit address system.

Fig. 3 illustrates the binary value results of plugging the address combinations of Fig. 1 into the sixteen binary logic functions (i.e., a-q) of Fig. 2.

Fig. 4 provides an example of how the sixteen binary logic functions (i.e., a-q) depicted in Fig. 2 can be used to represent certain arithmetic values (i.e., 20, 32, 16, and 4).

Fig. 5 provides another example of how the sixteen binary logic functions (i.e., a-q) depicted in Fig. 2 can be used to represent certain arithmetic values (i.e., "one," "log base two of ten," "log base two of *e*," and "*pi*").

Fig. 6 provides another example of how the sixteen binary logic functions (i.e., a-q) depicted in Fig. 2 can be used to represent certain arithmetic values (i.e., "log base ten of two," "log base *e* of two," "zero," and "don't care").

Fig. 7 illustrates how a three-bit address system could be used to identify the eight arithmetic values illustrated in Figs. 5 and 6.

Fig. 8 illustrates how the three-bit address system depicted in Fig. 7 could be rewritten as a two-bit address system with a most-significant bit (MSB).

Fig. 9 depicts one embodiment of the present invention that provides read-only data through the use of combinatorial logic.

Fig. 10 illustrates a six-bit address system.

Fig. 11 illustrates how the six-bit address system of Fig. 10 could be rewritten as a two-bit address system with four middle bits.

Fig. 12 depicts another embodiment of the present invention that provides read-only data through the use of combinatorial logic.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention provides a system and method for minimizing read-only data retrieval time by using combinatorial logic to provide read-only data to a requester. In the detailed description that follows, like element numerals are used to describe like elements illustrated in one or more figures.

Data is typically stored at address locations. A two-bit address system will first be described. Fig. 1 depicts a two-bit address system 10, which includes two address bits 110, 120 (i.e., x and y) that yield four address combinations (i.e., 00, 01, 10, and 11). Through the use of combinatorial logic, these two address bits 110, 120 (i.e., x and y) can be used to produce sixteen binary functions.

Fig. 2 illustrates the sixteen binary functions 220 in a binary function table 20 -- each one being represented by a separate and distinct reference character 210 (i.e., a-

q). More particularly, the sixteen logic functions 220 are made up of (1) zero, (2) one, (3) x, (4) y, and (5) x and/or y operated on by a plurality of logical functions that include OR (represented through the use of "|"), AND (represented through the use of "&"), XOR (represented through the use of "^"), and NOT (represented through the use of "~") (which can be used to produce NOR, NAND, and XNOR). Fig. 3 shows the binary value results 30 of plugging the address combinations of Fig. 1 into the logic functions 220 of Fig. 2.

From Figs. 2 and 3 it is clear that certain logic function results are independent of x 110 and y 120, whereas other logic function results are directly dependent on x 110 and/or y 120. For example, the logic function denoted by reference character "a" (i.e., function "0") produces a result of zero regardless of the value of x 110 or y 120. On the other hand, the logic function denoted by reference character "b" (i.e., function "x & y") produces a result of one only when x 110 and y 120 are both one. As another example, the logic function denoted by reference character "c" (i.e., function "x & (~y)") produces a result of one only when x 110 is one and y 120 is zero.

Fig. 4 shows how the these logic functions (denoted by "a-q") can be used to produce a desired result (e.g., read-only data). Specifically, two address bits 410 (i.e., x and y) are used to produce four address combinations (i.e., 00, 01, 10, and 11) and four corresponding read-only data 420 values (i.e., 20, 32, 16, and 4). It should be appreciated, however, that the specific read-only data values depicted in Figs. 4-11 (e.g., 20, 32, log base ten of two, etc.) are used to illustrate particular embodiments of the present invention, and are not intended to further limit the present invention. Therefore, any read-only data values generally known to those skilled in the art are within the spirit and scope of the present invention.

Referring back to Fig. 4, each read-only data 420 value (i.e., 20, 32, 16, 4) can have a corresponding binary representation. For example, the base ten data value "20" can be represent by the binary (i.e., base two) value "010100." As well, the base ten data value "32" can be represented by the binary value "100000," and so on. The end result is four binary values 430 that can be expressed in functional form. For example,

the most-significant bit (MSB) of each binary value is one only when x is zero and y is one. This is the same result as the logic function "e" (i.e., function "(~x) & y"), as depicted in Figs. 2 and 3. Thus, the MSB of the four binary values can be represented by the logic function "e." Similarly, the next most-significant bit (NMSB) is one only when y is zero. This is the same result as the logic function "k" (i.e., function "~y"), as depicted in Figs. 2 and 3. Thus, the NMSB of the four bit values can be represented by the logic function "k," and so on. By converting all six bits to logic functions, it becomes clear that the four binary values (and therefore the read-only data represented by these values) can be represented by the logic functions "ek-ajaa" 440. Once the logic functions 440 have been derived, combinatorial logic can be used to create a read-only memory system (see Fig. 9).

A similar method can be used to produce the results of the load constant instruction set (i.e., read-only data) for the Pentium processor. The instruction set for the Pentium processor includes seven read-only data values, namely: (1) one, (2) log base two of ten, (3) log base two of *e*, (4) *pi*, (5) log base ten of two, (6) log base *e* of two, and (7) zero. Fig. 5 depicts how two address bits 510 can be used to represent four (i.e., 520) of the seven read-only data values of the Pentium instruction set. More particularly, each read-only data 520 value can be represented in IEEE format -- that is, through a sign bit 530, eight exponent bits 540 (which represent two to the one-twenty-seventh-minus EXP 540 power), and twenty-three fraction bits 560. The base number (BN 550) (also called the integer bit or the implicit bit) for each binary value defaults to one, and thus does not need to be represented by a bit. Thus, the read-only data 520 value "one" is represented by "1.0" (i.e., 550, 560) times two to the zero power (i.e., one-twenty-seven minus EXP 540), which equals "one." It should be appreciated, however, that the method depicted in Fig. 5 for representing fractional and/or exponential read-only data values (i.e., the IEEE format) is used merely to illustrate one embodiment of the present invention, and is not intended to further limit the invention. Therefore, any binary representation of read-only data is within the spirit and scope of the present invention.

Referring back to Fig. 5, each bit of the four thirty-two bit values can be analyzed to determine which logic function produces an equivalent result. For example, the sign bit 530 is zero regardless of the value of x and y. This is the same result as the logic function "a" (function "0"), as depicted in Figs. 2 and 3. As well, the MSB of the exponent bits 540 is one whenever y is one. This is the same result as the logic function "f" (function "y"), as depicted in Figs. 2 and 3. The end result of such an analysis is that all four binary values (and therefore the read-only data) can be represented by the logic functions "a-fkkk-kkkl-fcgd-eabg-aceh-bhbb-fghh-add" 570 (i.e., the first set of logic functions). It should be appreciated that any logic function that is specified more than once (e.g., "d," in the logic functions 570), only needs to be computed once for the result to be used in multiple positions.

Similarly, Fig. 6 depicts how two address bits 610 can be used to produce the remaining three (i.e., 620) of the seven read-only data values of the Pentium instruction set (as well as one don't care - "DC"). As with Fig. 5, a sign bit 630, eight exponent bits 640, and twenty-three fraction bits 660 are used to represent the read-only data 620 values. As previously shown, each bit of the four binary values can be analyzed to determine which logic function produces an equivalent result. After analyzing each bit, it is clear that all four binary values (and therefore the read-only data) can be represented by the logic functions "b-bnnn-nnnj-bfmi-aiea-emea-aeai-aamm-aii" 670 (i.e., the second set of logic functions). Therefore, it is clear that two two-bit address systems (i.e., Figs. 5 and 6) can be used to produce the entire set of constants provided in the Pentium instruction set.

In order for a single system to produce the entire set of constants provided in the instruction set, that system would have to produce seven read-only data values. The number of read-only data values available for a k-bit address system is represent by $2^k$. In other words, a two-bit address system results in $2^2$ (or four) read-only data values; a three-bit address system results in $2^3$ (or eight) read-only data values; and so on. Therefore, in order to represent the seven read-only data values of the Pentium instruction set, a three-bit address system is needed.

Fig. 7 depicts a three-bit address system 70 capable of providing the eight read-only data 720 values of the Pentium instruction set. More particularly, the first four addresses (i.e., 000-011) correspond to the read-only data values depicted in Fig. 5, and the last four addresses (i.e., 100-111) correspond to the read-only data values depicted in Fig. 6. The similarity between Fig. 7 and Figs. 5 and 6 is perhaps better show in Fig. 8, where the most-significant bit (MSB) has been separated from the rest of the address -- in essence rewriting the three-bit address in one-bit-followed-by-two-bit address form.

By representing the three-bit address system in this fashion, it become clear that the MSB is used to select between the first four read-only data values and the last four read-only data values. For example, referring to Fig. 8, a zero for both x and y indicates either a read-only data value of one or log base ten of two -- the only difference being the value of the MSB. Therefore, depending on the MSB, the requester will either be provided with read-only data representing "one" or read-only data representing "log base ten of two" -- or in functional terms, the requester will either be provided with the result of the first set of logic functions (i.e., see Fig. 5, "a-fkkk-kkkl-fcgd-eabg-aceh-bhbb-fghh-add" 570) or the result of the second set of logic functions (i.e., see Fig. 6, "b-bnnn-nnnj-bfmi-aiea-emea-aeai-aamm-aii" 670).

Fig. 9 depicts an embodiment of the present invention that utilizes combinatorial logic to produce the read-only data depicted in Fig. 8. More particularly, a binary logic function device 910 is used, along with a multiplexer 920, to produce a specified read-only data value 930. Specifically, two address bits 714 (i.e., x and y) are provided to the binary logic function device 910. The binary logic function device 910 uses the two address bits 714 and predetermined logic functions (i.e., the functions that make up the first and second set of logic functions 570, 670) to produce binary values. The binary values are then provided to a multiplexer 920. The multiplexer 920, in turn, uses the MSB to select which binary value is actually provided to the requester. It should be appreciated that the term requester is being used in its broad sense, and any person, program, circuit, or device requesting read-only data is within the spirit and scope of the

present invention. Furthermore, it should be appreciated that the multiplexer 920 depicted in Fig. 9 is not required to practice the present invention, and is only needed in a three-bit (or more) address system. For example, a multiplexer is not needed to produce the read-only data depicted in Fig. 4 (i.e., a two-bit address system).

It should be appreciated, however, that the present invention is not limited to a three-address system. For example, as depicted in Fig. 10, a DES encryption algorithm utilizes a six-bit address system for providing sixty-four read-only memory values. Fig. 11 depicts how this six-bit address system can be rewritten in two-bit address form. More particularly, two of the six bits (e.g., bits "0" and "5") (the particular two being of no significance) are mapped on one axis, and the remaining bits (e.g., bits "1-4") are mapped on the other axis. This allows the read-only data values to be represented by logic functions. For example, when bits "1-4" are 0000, the read-only data values are either 1110, 0000, 0100, or 1111, depending on the value of bits "0" and "5." Thus, the MSB of these values is one whenever both x and y are both one or both zero. This is the same result as the logic function "j" (i.e., function "~(x ^ y)"). This process can be continued to determine that the four bit values (and therefore the read-only data) is represented by the logic functions of "jljb" 1130. By doing this for the entire six-bit address system, a series of logic functions (i.e., 1130-1160) representing the read-only data values can be produced.

Fig. 12 depicts an embodiment of the present invention that utilizes combinatorial logic to produce the read-only data depicted in Fig. 11. More particularly, a binary logic function device 1202 is used, along with a plurality of multiplexers 1210-1218, to produce a specified read-only data value 1220. Specifically, two address bits 1120 (i.e., bits "0" and "5") are provided to the binary logic function device 1202. The binary logic function device uses the two address bits 1120 and predetermined logic functions (i.e., the logic function 1130-1160) to produce binary values. The binary values are then provided to the four multiplexers 1210-1216. Address bits "3-4", which are first decoded through a decoder 1204, are provided to the four multiplexers 1210-1216 to determine (or select) which set of binary values are provided to a fifth multiplexer 1218. Address

bits "1-2", which are first decoded through a decoder 1206, are provided to the fifth multiplexer 1218 to determine (or select) which binary value is the read-only data requested (i.e., 1220). It should be appreciated that the number of multiplexers 1210-1218 depicted in Fig. 12 is not intended to limit the present invention, but merely to illustrate one embodiment of the present invention. Furthermore, it should be appreciated that the predetermined logic functions do not necessarily include every logic function depicted in Fig. 2. Utilizing only a subset of the logic functions depicted in Fig. 2 is considered within the spirit and scope of the present invention.

Having thus described a preferred embodiment of a system and method for minimizing read-only data retrieval time through the use of combinatorial logic, it should be apparent to those skilled in the art that certain advantages of the system have been achieve. It should also be appreciated that various modifications, adaptations, and alternative embodiments thereof may be made within the scope and spirit of the present invention. The invention is further defined by the following claims.